

D Intrinsic Functions and Procedures

The following abbreviations and typographic conventions are used in this appendix.

Argument type and result type:

I	Integer
R	Real
C	Complex
N	Numeric (any of integer, real, complex)
L	Logical
P	Pointer
T	Target
DP	Double precision
Char	Character, length = 1.
S	Character

Class

E	Elemental function
I	Inquiry function
T	Transformational function
S	Subroutine

See Chapter 14 for more information on these classifications.

Arguments in italics

ALL(Mask,Dim)

are optional arguments, i.e., *Dim* may be omitted in the example above.

Double precision

Before Fortran 90 if you required real variables to have greater precision than the default real then the only option available was to declare them as double precision. With the introduction of kind types with Fortran 90 the use of double precision

declarations is not recommended, and instead real entities with a kind type offering more than the default precision should be used.

Kind optional argument

There are several functions that have an optional argument Kind, e.g., AINT(A,Kind). If Kind is absent the result is the same kind type as the first argument, in this case A. If Kind is present the result has the kind type specified by this argument.

Result type

When the result type is the same as the argument type then the result is not just the same type as the argument but also the same kind.

Miscellaneous rules

When the argument is Back it is of logical type.

When the argument is Count_Rate, Count_Max, Dim, Kind, Len, Order, N_Copies, Shape, Shift, Values it is of integer type.

When the argument is Mask it is of logical type.

When the argument is Target it is of pointer or target type.

ABS(A)

Yields the absolute value unless A is complex; see below.

Argument: A **Type:** N

Result: As argument **Class:** E

Note: If A is complex (x,y) then the functions returns $\sqrt{x^2 + y^2}$

Example: R1=ABS(A)

ACHAR(I)

Returns character in the ASCII character set.

Argument: I **Type:** I

Result: Char **Class:** E

Example: C=ACHAR(I)

ACOS(X)

Arccosine (inverse cosine).

Argument: X **Type:** R

Result: As argument **Class:** E

Note: $|x| \leq 1$

Example: Y=ACOS(X)

ADJUSTL(String)

Adjust string left, removing leading blanks and inserting trailing blanks.

Argument: String **Type:** S

Result: As argument **Class:** E

Example: S=ADJUSTL(S)

ADJUSTR(String)

Adjust string right, removing trailing blanks and inserting leading blanks.

Argument: String **Type:** S

Result: As argument **Class:** E

Example: S=ADJUSTR(S)

AIMAG(Z)

Imaginary part of complex argument.

Argument: Z **Type:** C

Result: As argument **Class:** E

Example: Y=AIMAG(Z)

AINT(A,Kind)

Truncation.

Argument: A **Type:** R

Result: As A **Class:** E

Argument: Kind **Type:** I

Example: Y=AINT(Z) and when Z=0.3 Y=0, when Z=2.73 Y=2.0, when Z=-2.73 Y=-2.0

ALL(Mask,Dim)

Determines whether all values are true in Mask along dimension Dim.

Argument: Mask **Type:** L

Result: L**Class:** T

Note: Dim must be a scalar in the range $1 \leq Dim \leq n$ where n is the rank of Mask. The result is scalar if Dim is absent or Mask has rank 1. Otherwise it works on the dimension Dim of Mask and the result is an array of rank $n-1$.

Example: T=ALL(M)**ALLOCATED(Array)**

Returns true if array is allocated.

Argument: Array**Type:** Any**Result:** L**Class:** I

Note: Array must be declared with the ALLOCATABLE attribute.

Example: IF (ALLOCATED(Array)) THEN ...**ANINT(A,Kind)**

Rounds reals, i.e., returns nearest whole number.

Argument: A**Type:** R**Result:** As A**Class:** E**Example:** Z=ANINT(A), if A = 5.63 Z = 6, if A=-5.7 Z = -6.0**ANY(Mask,Dim)**

Determines whether any value is true in Mask along dimension Dim.

Argument: Mask**Type:** L**Result:** L**Class:** T

Note: Mask must be an array. The result is a scalar if Dim is absent or if Mask is of rank 1. Otherwise it works on the dimension Dim of Mask and the result is an array of rank $n-1$.

Example: T=ANY(A)**ASIN(X)**

Arcsine.

Argument: X**Type:** R**Result:** As argument**Class:** E**Example:** Z=ASIN(X)

ASSOCIATED(Pointer,Target)

Returns the association status of the pointer.

Argument: Pointer	Type: P
Result: L	Class: I

Note:

1. If Target is absent then the result is true if POINTER is associated with a target, otherwise false.
2. If Target is present and is a target, the result is true if Pointer is currently associated with Target and false if it is not.
3. If Target is present and is a pointer, the result is true if both Pointer and Target are currently associated with the same target, and is false otherwise. If either Pointer or Target is disassociated the result is false.

Example: T=ASSOCIATED(P)

ATAN(X)

Arctangent.

Argument: X	Type: R
Result: As argument	Class: E

Example: Z=ATAN(X)

ATAN2(Y,X)

Arctangent of Y / X.

Argument: Y	Type: R
Result: As arguments	Class: E

Example: Z=ATAN2(Y,X)

BIT_SIZE(I)

Returns the number of bits, as defined by the numeric model for integer numbers in Chapter 8.

Argument: I	Type: I
Result: As argument	Class: I

Example: N_Bits=SIZE(I)

BTEST(I,Pos)

Returns true if the bit is set in the integer argument at the position given by the second argument.

Argument: I	Type: I
Result: L	Class: E

Example: T=BTEST(I,Pos)

CEILING(A,Kind)

Returns the smallest integer greater than or equal to the argument.

Argument: A	Type: R
Result: I	Class: E

Note:

If kind is present the result has the kind type parameter Kind.

Otherwise the result is of type default integer.

Example: I=CEILING(A) If A=12.21 then I=13, if A=-3.16 then I=-3

CHAR(I,Kind)

Returns the character in a given position in the processor collating sequence associated with the specified kind type parameter. Normally ASCII.

Argument: I	Type: I
Result: CHAR	Class: E

Example: C=CHAR(65) and for the ASCII character set C='A'.

CMPLX(X,Y,Kind)

Converts to complex from integer, real and complex.

Argument: X	Type: N
Result: C	Class: E

Note:

1. If X is complex and Y is absent it is as if Y were present with the value AIMAG(X).
2. If X is not complex and Y is absent, it is as if Y were present with the value 0.

Example: Z=CMPLX(X,Y)

CONJG(Z)

Conjugate of a complex argument.

Argument: Z **Type:** C

Result: As Z **Class:** E

Example: Z1=CONJG(Z)

COS(X)

Cosine.

Argument: X **Type:** R, C

Result: As argument **Class:** E

Note: The arguments of all trigonometric functions should be in radians, not degrees.

Example: A=COS(X)

COSH(X)

Hyperbolic cosine.

Argument: X **Type:** R

Result: As argument **Class:** E

Example: Z=COSH(X)

COUNT(Mask,Dim)

Returns the number of true elements in Mask along dimension Dim.

Argument: Mask **Type:** L

Result: I **Class:** T

Note: Dim must be a scalar in the range $1 \leq Dim \leq n$, where n is the rank of Mask. The result is scalar if Dim is absent or Mask has rank 1. Otherwise it works on the dimension Dim of Mask and the result is an array of rank $n-1$.

Example: N=COUNT(A)

CPU_TIME(Time)

Returns the processor time.

Argument: Time **Type:** R

Result: N/A **Class:** S

Example: CALL CPU_TIME(Time)

CSHIFT(Array,Shift,Dim)

Circular shift on a rank 1 array or rank 1 sections of higher-rank arrays.

Argument: Array **Type:** Any

Result: As Array **Class:** T

Note: Array must be an array, Shift must be a scalar if Array has rank 1, otherwise it is an array of rank $n-1$, where n is the rank of Array. Dim must be a scalar with a value in the range $1 \leq Dim \leq n$.

Example: Array=CSHIFT(Array,10)

DATE_AND_TIME(Date,Time,Zone,Values)

Returns the current date and time (compatible with ISO 8601:1988).

Argument: Date **Type:** S

Result: N/A **Class:** S

Time and Zone are of type S.

Note:

1. Date is optional and must be scalar and 8 characters long in order to return the complete value of the form CCYYMMDD, where CC is the century, YY is the year, MM is the month and DD is the day. It is INTENT(OUT).
2. Time is optional and must be scalar and 10 characters long in order to return the complete value of the form hhmmss.sss where hh is the hour, mm is the minutes and ss.sss is the seconds and milliseconds. It is INTENT(OUT).
3. Zone is optional and must be scalar and must be 5 characters long in order to return the complete value of the form hhmm where hh and mm are the time differences with respect to Coordinated Universal Time in hours and minutes. It is INTENT(OUT).
4. Values is optional and a rank 1 array of size 8. It is INTENT(OUT). The values returned are as follows:
 Values(1) = the year
 Values(2) = the month
 Values(3) = the day
 Values(4) = the time with respect to Coordinated Universal Time in minutes.
 Values(5) = the hour (24 hour clock)
 Values(6) = the minutes
 Values(7) = the seconds
 Values(8) = the milliseconds in the range 0–999.

Example: CALL DATE_TIME(D,T,Z,V)

DBLE(A)

Converts to double precision from integer, real, and complex

Argument: A **Type:** N

Result: DP **Class:** E

Example: D=DBLE(A)

DIGITS(X)

Returns the number of significant digits of the argument as defined in the numeric models for integer and reals in Chapter 8.

Argument: X **Type:** I,R

Result: I **Class:** I

Example: I=DIGITS(X)

DIM(X,Y)

Returns first argument minus minimum of the two arguments: X -MIN(X,Y).

Argument: X **Type:** I

Result: As arguments **Class:** E

Example: Z=DIM(X,Y)

DOT_PRODUCT(Vector_1,Vector_2)

Performs the mathematical dot product of two rank 1 arrays.

Argument: Vector_1 **Type:** Nt

Result: As arguments **Class:** T

Vector_2 is as Vector_1.

Note:

1. If Vector_1 is of type integer or real the result has the value SUM(Vector_1*Vector_2).
2. If Vector_1 is complex the result has the value SUM(CONJG(Vector_1)*Vector_2).
3. If Vector_1 is logical the result has the value ANY(Vector_1 .AND. Vector_2).

Example: A=DOT_PRODUCT(X,Y)

DPROD(X,Y)

Double precision product of two reals.

Argument: X	Type: R
Result: DP	Class: E

Example: D=DPROD(X,Y)

EOSHIFT(Array,Shift,Boundary,Dim)

End of shift of a rank 1 array or rank 1 section of a higher-rank array.

Argument: Array	Type: Any
Result: As Array	Class: T

Boundary is as Array.

Note: Array must be an array, Shift must be a scalar if Array has rank 1, otherwise it is an array of rank $n-1$, where n is the rank of Array. Boundary must be scalar if Array has rank 1, otherwise it must be either scalar or of rank $n-1$. Dim must be a scalar with a value in the range $1 \leq Dim \leq n$.

Example: A=EOSHIFT(A,Shift)

EPSILON(X)

Smallest difference between two reals of that kind. See Chapter 8 and real numeric model.

Argument: X	Type: R
Result: As argument	Class: I

Example: Tiny=EPSILON(X)

EXP(X)

Exponential, e^x .

Argument: X	Type: R, C
Result: As argument	Class: E

Example: Y=EXP(X)

EXPONENT(X)

Returns the exponent component of the argument. See Chapter 8 and the real numeric model.

Argument: X	Type: R
Result: I	Class: E

Example: I=EXPONENT(X)

FLOOR(A, Kind).

Returns the greatest integer less than or equal to the argument

Argument: A

Type: R

Result: I

Class: E

Note:

If kind is present the result has the kind type parameter Kind, otherwise the result is of type default integer.

Example: I=FLOOR(A) and when A=5.2 I has the value 5, when A=-9.7 I has the value -10

FRACTION(X)

Returns the fractional part of the real numeric model of the argument See Chapter 8 and the real numeric model.

Argument: X

Type: R

Result: As X

Class: E

Example: F=FRACTION(X)

HUGE(X)

Returns the largest number for the kind type of the argument. See Chapter 8 and the real and integer numeric models.

Argument: X

Type: I,R

Result: As argument

Class: I

Example: H=HUGE(X)

IACHAR(C)

Returns the position of the character argument in the ASCII collating sequence.

Argument: C

Type: Char

Result: I

Class: E

Example: I=IACHAR('A') returns the value 65.

IAND(I,J)

Performs a logical AND on the arguments.

Argument: I

Type: I

Result: As arguments

Class: E

Example: `K=IAND(I,J)`

IBCLR(I,Pos)

Clears one bit of the argument to zero.

Argument: I **Type:** I

Result: As I **Class:** E

Note: $0 \leq Pos < BIT_SIZE(I)$

Example: `I=IBCLR(I,Pos)`

IBITS(I,Pos,Len)

Returns a sequence of bits.

Argument: I **Type:** I

Result: As I **Class:** E

Note: $0 \leq Pos$ and $(Pos + Len) \leq BIT_SIZE(I)$ and $Len \geq 0$.

Example: `Slice=IBITS(I,Pos,Len)`

IBSET(I,Pos)

Sets one bit of the argument to one.

Argument: I **Type:** I

Result: As I **Class:** E

Note: $0 \leq Pos < BIT_SIZE(I)$.

Example: `I=IBSET(I,Pos)`

ICHAR(C)

Returns the position of a character in the processor collating sequence associated with the kind type parameter of the argument. Normally the position in the ASCII collating sequence.

Argument: C **Type:** CHAR

Result: I **Class:** E

Example: `I=ICHAR('A')` would return the value 65 for the ASCII character set.

IEOR(I,J)

Performs an exclusive OR on the arguments.

Argument: I **Type:** I

Result: As I**Class:** E**Example:** I=IEOR(I,J)**INDEX(String,Substring,Back)**

Locates one substring in another, i.e., returns position of Substring in character expression String.

Argument: String**Type:** S**Result:** I**Class:** E

Substring is of type S.

Note:

1. If Back is absent or present with the value .FALSE. then the function returns the start position of the first occurrence of the substring. If LEN(Substring) = 0 then one is returned.
2. If Back is present with the value .TRUE. then the function returns the start position of the last occurrence of the substring. If LEN(Substring) = 0 then the value (LEN(String) + 1) is returned.
3. If the substring is not found the result is zero.
4. If LEN(String) < LEN(Substring) the result is zero.

Example:

Where=INDEX(' Hello world Hello','Hello')

The result 2 is returned.

Where=INDEX(' Hello world Hello','Hello',.TRUE.)

The result 14 is returned.

INT(A,Kind)

Converts to integer from integer, real, and complex.

Argument: A**Type:** N**Result:** I**Class:** E**Example:** I=INT(F)**IOR(I,J)**

Performs an inclusive OR on the arguments.

Argument: I**Type:** I**Result:** As I**Class:** E**Example:** I=IOR(I,J)

ISHFT(I,Shift)

Performs a logical shift. The bits of I are shifted by Shift positions.

Argument: I

Type: I

Result: As I

Class: E

Note: $|Shift| \leq BIT_SIZE(I)$

Example: I=ISHIFT(I,Shift).

ISHFTC(I,Shift,Size)

Performs a circular shift of the rightmost bits. The Size rightmost bits of I are circularly shifted by Shift positions.

Argument: I

Type: I

Result: I

Class: E

Note:

$|Shift| \leq Size$

$0 \leq Size \leq BIT_SIZE(I)$.

If Size is absent it is as if it were present with the value of BIT_SIZE(I).

If Shift is positive the shift is to the left.

If Shift is negative the shift is to the right.

If Shift is zero no shift is performed.

Example: I=ISHFTC(I,Shift,Size)

KIND(X)

Returns the KIND type parameter of the argument.

Argument: X

Type: Any

Result: I

Class: I

Example: I=KIND(X)

LBOUND(Array,Dim)

Returns the lower bounds for each dimension of the array argument or a specified lower bound.

Argument: Array

Type: Any

Result: I

Class: I

Note:

$1 \leq Dim \leq n$, where n is the rank of Array. The result is scalar if Dim is present oth-

erwise the result is an array of rank 1 and size n .

The result is scalar if Dim is present, otherwise a rank 1 array and size n .

Example: I=LBOUND(Array)

LEN(String)

Length of a character entity.

Argument: String **Type:** S

Result: I **Class:** I

Example: I=LEN(String)

LEN_TRIM(String)

Length of character argument less the number of trailing blanks.

Argument: String **Type:** S

Result: I **Class:** E

Example: I=LEN_TRIM(String)

LGE(String_1,String_2)

Lexically greater than or equal to and this is based on the ASCII collating sequence.

Argument: String_1 **Type:** S

Result: L **Class:** E

String_2 is of type S.

Example: L=LGE(S1,S2)

LGT(String_1,String_2)

Lexically greater than and this is based on the ASCII collating sequence.

Argument: String_1 **Type:** S

Result: L **Class:** E

Example: L=LGT(S1,S2)

LLE(String_1,String_2)

Lexically less than or equal to and this is based on the ASCII collating sequence.

Argument: String_1 **Type:** S

Result: L **Class:** E

String_2 is of type S.

Example: L=LLE(S1,S2)

LLT(String_1,String_2)

Lexically less than and this is based on the ASCII collating sequence.

Argument: String_1 **Type:** S

Result: L **Class:** E

Example: L=LLT(S1,S2)

LOG(X)

Natural logarithm, $\log_e x$.

Argument: X **Type:** R, C

Result: As argument **Class:** E

Example: Y=LOG(X)

LOG10(X)

Common logarithm, $\log_{10}$.

Argument: X **Type:** R

Result: As argument **Class:** E

Example: Y=LOG10(X)

LOGICAL(L,Kind)

Converts between different logical kind types, i.e., performs a type cast.

Argument: L **Type:** L

Result: L **Class:** E

Example: L=LOGICAL(K,Kind)

MATMUL(Matrix_1,Matrix_2)

Performs mathematical matrix multiplication of the array arguments.

Argument: Matrix_1 **Type:** N,L

Result: As arguments **Class:** T

Matrix_2 is as Matrix_1.

Note:

1. Matrix_1 and Matrix_2 must be arrays of rank 1 or 2. If Matrix_1 is of numeric

type so must Matrix_2.

2. If Matrix_1 has rank 1, Matrix_2 must have rank 2.
3. If Matrix_2 has rank 1, Matrix_1 must have rank 2.
4. The size of the first dimension of Matrix_2 must equal the size of the last dimension of Matrix_1.
5. If Matrix_1 has shape (n,m) and Matrix_2 has shape (m,k) the result has shape (n,k) .
6. If Matrix_1 has shape (m) and Matrix_2 has shape (m,k) the result has shape (k) .
7. If Matrix_1 has shape (n,m) and Matrix_2 has shape (m) the result has shape (n) .

Example: R=MATMUL(M_1,M_2)

MAX(A1,A2,A3,...)

Returns the largest value.

Argument: A1 **Type:** I,R

Result: As arguments **Class:** E

A2, A3,... are as A1.

Example: A=MAX(A1,A2,A3,A4)

MAXEXPONENT(X)

Returns the maximum exponent. See Chapter 8 and numeric models.

Argument: X **Type:** R

Result: I **Class:** I

Example: I=MAXEXPONENT(X)

MAXLOC(ARRAY,Dim,Mask)

Determine the location of the first element of Array having the maximum value of the elements identified by Mask if present.

Argument: Array **Type:** I,R

Result: I **Class:** T

Note:

0. Normally in Fortran 95 if you omit an optional argument you must use keywords for the rest. This intrinsic breaks this rule and DIM can be omitted and it is not necessary to use a keyword with Mask.

1. Array must be an array.
2. Mask must be conformable with Array
3. The result is an array of rank 1 and of size equal to the rank of Array.

4. If *Dim* is present the result is an array of the rank of *Array* reduced by one and with the shape of *Array* without the dimension *Dim*.

Example:

$A = (/5,6,7,8/)$

$I = \text{MAXLOC}(A)$

is (4), which is the subscript of the location of the first occurrence of the maximum value in the rank 1 array.

$$\text{If } A = \begin{pmatrix} 1 & 8 & 5 \\ 9 & 3 & 6 \\ 4 & 2 & 7 \end{pmatrix}$$

$I = \text{MAXLOC}(A, \text{dim}=1)$

is (2,1,3) returning the position of the largest in each column.

$I = \text{MAXLOC}(A, \text{dim}=2)$

is (2,1,3) returning the position of the largest in each row.

MAXVAL(Array,Dim,Mask)

Returns the maximum value of the elements of *Array* along dimension *Dim* corresponding to the true elements of *Mask*.

Argument: *Array* **Type:** I,R

Result: As argument **Class:** T

Note:

$1 \leq \text{Dim} \leq n$, where n is the rank of *Array*. The result is scalar if *Dim* is absent, or *Array* has rank 1. Otherwise the result is an array of rank $n-1$.

If *Array* has size zero then the result is the largest negative number supported by the processor for the corresponding type and kind of *Array*.

Example:

$\text{MAXVAL}(/1,2,3/)$ returns the value 3.

$\text{MAXVAL}(C, \text{MASK}=C < 0.0)$ returns the maximum of the negative elements of *C*.

$$\text{For } B = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$

$\text{MAXVAL}(B, \text{DIM}=1)$ returns (2,4,6)

$\text{MAXVAL}(B, \text{DIM}=2)$ returns (5,6)

MERGE(True,False,Mask)

Chooses alternative values according to the value of a mask.

Argument: True **Type:** Any

Result: As True **Class:** E

Example: for

For $True = \begin{pmatrix} 2 & 6 & 10 \\ 4 & 8 & 12 \end{pmatrix}$, $False = \begin{pmatrix} 1 & 5 & 9 \\ 3 & 7 & 11 \end{pmatrix}$ and $Mask = \begin{pmatrix} T & F & T \\ F & T & F \end{pmatrix}$

The result is $\begin{pmatrix} 2 & 5 & 10 \\ 3 & 8 & 11 \end{pmatrix}$

MIN(A1,A2,A3,...)

Chooses the smallest value.

Argument: A1 **Type:** I, R

Result: As arguments **Class:** E

Example: Y=MIN(X1,X2,X3,X4,X5)

MINEXPONENT(X)

Returns the minimum exponent. See Chapter 8 and numeric models.

Argument: X **Type:** R

Result: I **Class:** I

Example: I=MINEXPONENT(X)

MINLOC(Array,Dim,Mask)

Determine the location of the first element of Array having the minimum value of the elements identified by Mask.

Argument: Array **Type:** I,R

Result: I **Class:** T

Note:

0. Normally in Fortran 95 if you omit an optional argument you must use keywords for the rest. This intrinsic breaks this rule and Dim can be omitted and it is not necessary to use a keyword with Mask.

1. Array must be an array.
2. Mask must be conformable with Array.
3. The result is an array of rank 1 and of size equal to the rank of Array.

4. If DIM is present the result is an array of the rank of Array reduced by one and with the shape of Array without the dimension DIM.

Example: I=MINLOC(Array)

In the above example if Array is a rank 2 array of shape (5,10) and the smallest value is in position (2,1) then the result is the rank 1 array I with shape (2) and I(1)=2 and I(2)=1.

See MAXLOC for further examples.

MINVAL(Array,Dim,Mask)

Returns the minimum value of the elements of Array along dimension Dim corresponding to the true elements of Mask.

Argument: Array **Type:** I,R

Result: As Array **Class:** T

Note: $1 \leq \text{Dim} \leq n$, where n is the rank of Array. The result is scalar if Dim is absent, or Array has rank 1. Otherwise the result is an array of rank $n-1$.

If Array has size zero then the result is the largest negative number supported by the processor for the corresponding type and kind of Array.

Example:

MINVAL((/1,2,3/)) returns the value 1.

MINVAL(C,MASK=C > 0.0) returns the minimum of the positive elements of C.

For $B = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$

MINVAL(B,DIM=1) returns (1,3,5).

MINVAL(B,DIM=2) returns (1,2).

MOD(A,B)

Returns the remainder when first argument divided by second.

Argument: A **Type:** I, R

Result: As arguments **Class:** E

Note: If B=0 the result is processor dependent. For $B \neq 0$ the result is $A - \text{INT}(A/B) * B$.

Example: R=MOD(A,B)

If A=8 and B=5 then R=3

If A=-8 and B=5 then R=-3

If A=8 and B=-5 then R=3
 If A=-8 and B=-5 then R=-3

MODULO(A,B)

Returns the modulo of the arguments.

Argument: A **Type:** I,R

Result: As A **Class:** E

Note:

1. If B=0 then the result is processor dependent.
2. Integer A
 The result is R where $A = Q * B + R$ and Q is integer
 for $B > 0$, $0 \leq R < B$
 for $B < 0$, $B < R \leq 0$
3. Real A
 The result is $A - \text{FLOOR}(A/B) * B$.

Example: R=MODULO(A,B)

If A=8 and B=5 then R=3
 If A=-8 and B=5 then R=2
 If A=8 and B=-5 then R=-2
 If A=-8 and B=-5 then R=-3

MVBITS(From,F_Pos,Len,To,T_Pos)

Copies a sequence of bits from one data object to another.

Argument: From **Type:** I

Result: N/A **Class:** S

All arguments are of integer type.

Note:

From must be INTENT(IN).
 F_Pos must be INTENT(IN), $F_Pos \geq 0$, $F_Pos + Len \leq \text{BIT_SIZE}(\text{From})$.
 Len must be INTENT(IN), $Len \geq 0$.
 To must be INTENT(INOUT).
 T_Pos must be INTENT(IN), $T_Pos \geq 0$, $T_Pos + Len \leq \text{BIT_SIZE}(\text{To})$.

Example: CALL MVBITS(F,FP,L,T,TP)**NEAREST(X,Next)**

Returns the nearest different number. See Chapter 8 and the real numeric model.

Argument: X **Type:** R

Result: As X **Class:** E

Next is of type R.

Example: N=NEAREST(X,Next)

NINT(A,*Kind*)

Yields nearest integer.

Argument: A **Type:** RI

Result: I **Class:** E

Note:

1. $A > 0$, the result is $\text{INT}(A+0.5)$.
2. $A \leq 0$, the result is $\text{INT}(A-0.5)$.

Example: I=NINT(X)

NOT(I)

Returns the logical complement of the argument.

Argument: I **Type:** I

Result: As I **Class:** E

Example: I=NOT(I)

NULL(*Mold*)

Returns a disassociated pointer.

Argument: Mold **Type:** P

Result: As argument **Class:** T

Note:

If the argument Mold is present the result is the same as Mold.

Otherwise it is determined by context.

Example: REAL , POINTER :: P=>NULL()

PACK(Array,Mask,*Vector*)

Packs an array into an array of rank 1, under the control of a mask.

Argument: Array **Type:** Any

Result: As Array **Class:** T

Note:

1. Array must be an array.
2. Mask be conformable with Array.
3. Vector must have rank 1 and have at least as many elements as there are TRUE elements in Mask.
4. If Mask is scalar with the value TRUE. Vector must have at least as many elements as there are in Array.
5. The result is an array of rank 1.
6. If Vector is present the result size is that of Vector.
7. If Vector is not present the result size is t , the number of TRUE elements in Mask, unless Mask is scalar with a value TRUE in which case the result size is the size of Array.

Example: R=PACK(A,M)

PRECISION(X)

Returns the decimal precision of the argument. See Chapter 8 and numeric models.

Argument: X **Type:** R, C

Result: I **Class:** I

Example: I=PRECISION(X)

PRESENT(A)

Returns whether an optional argument is present.

Argument: A **Type:** Any

Result: L **Class:** I

Note: A must be an optional argument of the procedure in which the PRESENT function reference appears.

Example: IF (PRESENT(X)) THEN ...

PRODUCT(Array,Dim,Mask)

The product of all of the elements of Array along the dimension Dim corresponding to the TRUE elements of Mask.

Argument: Array **Type:** N

Result: As Array **Class:** T

Note:

1. Array must be an array.
2. $1 \leq \text{Dim} \leq n$ where n is the rank of Array.
3. Mask must be conformable with Array.

4. Result is scalar if Dim is absent, or Array has rank 1, otherwise the result is an array of rank $n-1$.

Example:

1. `PRODUCT((/1,2,3/))` the result is 6.
2. `PRODUCT(C,Mask=C > 0.0)` forms the product of the positive elements of C.

3. If $B = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$

`PRODUCT(B,DIM=1)` is (2,12,30) and
`PRODUCT(B,DIM=2)` is (15,48)

RADIX(X)

Returns the base of the numeric argument. See Chapter 8 and numeric models.

Argument: X **Type:** I,R

Result: I **Class:** I

Example: Base=`RADIX(X)`

RANDOM_NUMBER(X)

Returns one pseudorandom number or an array of pseudorandom numbers from the uniform distribution over the range $0 \leq x < 1$

Argument: X **Type:** R

Result: N/A **Class:** S

Note: X is `INTENT(OUT)`.

Example: `CALL RANDOM_NUMBER(X)`

RANDOM_SEED(Size,Put,Get)

Restarts (seeds) or queries the pseudorandom generator used by `RANDOM_NUMBER`.

Argument: Size **Type:** I

Result: N/A **Class:** S

All arguments are of integer type.

Note:

1. Size is `INTENT(OUT)`. It is set to the number N of integers that the processor uses to hold the value of the seed.
2. Put is `INTENT(IN)`. It is an array of rank 1 and size $\geq N$. It is used by the processor to set the seed value.

3. Get is INTENT(OUT). It is an array of rank 1 and size $\geq N$. It is set by the processor to the current value of the seed.

Example: CALL RANDOM_SEED

RANGE(X)

Returns the decimal exponent range of the real argument. See Chapter 8 and the numeric model representing the argument.

Argument: X **Type:** N

Result: I **Class:** I

Example: I=RANGE(N)

REAL(A,Kind)

Converts to real from integer, real or complex.

Argument: A **Type:** N

Result: R **Class:** E

Example: X=REAL(A)

REPEAT(String,N_Copies)

Concatenates several copies of a string.

Argument: String **Type:** S

Result: S **Class:** T

Example: New_S=REPEAT(S,10)

RESHAPE(Source,Shape,Pad,Order)

Constructs an array of a specified shape from the elements of a given array.

Argument: Source **Type:** Any

Result: As Source **Class:** T

Note:

1. Source must be an array. If Pad is absent or of size zero the size of Source must be $\geq \text{PRODUCT}(\text{Shape})$.
2. Shape must be a rank 1 array and $0 \leq \text{size} < 8$.
3. Pad must be an array.
4. Order must have the same shape as Shape and its value must be a permutation of (1,2,...,n) where n is the size of Shape. If absent it is as if it were present with the value (1,2,...,n).
5. The result is an array of shape, Shape.

Example:

RESHAPE((/1,2,3,4,5,6/),(/2,3/)) has the value $\begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$

RESHAPE((/1,2,3,4,5,6/) , (/2,4/) , (/0,0/) , (/2,1/)) has the value $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 0 & 0 \end{pmatrix}$

RRSPACING(X)

Returns the reciprocal of the relative spacing of model numbers near the argument value. See Chapter 8 and the real numeric model.

Argument: X **Type:** R

Result: As X **Class:** E

Example: Z=RRSPACING(X)

SCALE(X,I)

Returns $X * b^I$ where b is the base in the model representation of X. See Chapter 8 and the real numeric model.

Argument: X **Type:** R

Result: As X **Class:** E

I is of integer type.

Example: Z=SCALE(X,I)

SCAN(String,Set,Back)

Scans a string for any one of the characters in a set of characters.

Argument: String **Type:** S

Result: I **Class:** E

Note:

1. The default is to scan from the left, and will only be from the right when Back is present and has the value TRUE.
2. Zero is returned if the scan fails.

Example: W=SCAN(String,Set)

SELECTED_INT_KIND(R)

Returns a value of the kind type parameter of an integer data type that represents all integer values n with $-10^R < n < 10^R$

Argument: R **Type:** I

Result: I**Class:** T**Note:**

R must be scalar.

If a kind type parameter is not available then the value -1 is returned.

Example: `I=SELECTED_INT_KIND(2)`

SELECTED_REAL_KIND(*P*,*R*)

Returns a value of the kind type parameter of a real data type with decimal precision of at least *P* digits and a decimal exponent range of at least *R*.

Argument: *P* and *R***Type:** I**Result:** I**Class:** T**Note:**

1. *P* and *R* must be scalar.
2. The value -1 is returned if the precision is not available, the value -2 if the exponent range is not available, and -3 if neither is available.

Example: `I=SELECTED_REAL_KIND(P,R)`

SET_EXPONENT(*X*,*I*)

Returns the model number whose fractional part is the fractional part of the model representation of *X* and whose exponent part is *I*.

Argument: *X***Type:** R**Result:** As *X***Class:** E

I is of integer type.

Example: `Exp_Part=SET_EXPONENT(X,I)`

SHAPE(*Source*)

Returns the shape of the array argument or scalar.

Argument: *Source***Type:** Any**Result:** I**Class:** I**Note:**

1. *Source* may be array valued or scalar. It must not be a pointer that is disassociated or an allocatable array that is not allocated. It must not be an assumed-size array.
2. The result is an array of rank 1 whose size is equal to the rank of *Source*.

Example: `S=SHAPE(A(2:5,-1:1))` yields `S=(4,3)`

SIGN(A,B)

Absolute value of A times the sign of B.

Argument: A **Type:** I, R

Result: As A **Class:** E

Note:

In the special case where B is zero normally the result would have the value ABS(A), but if B is one of the real kind types and the processor is able to distinguish between plus zero and minus zero then the result is ABS(A) if B is plus zero and the result is $-ABS(A)$ if B is minus zero.

B is as A.

Example: A=SIGN(A,B)

SIN(X)

Sine.

Argument: X **Type:** R, C

Result: As argument **Class:** E

Note: The argument is in radians.

Example: Z=SIN(X)

SINH(X)

Hyperbolic sine.

Argument: X **Type:** R

Result: As argument **Class:** E

Example: Z=SINH(X)

SIZE(Array,Dim)

Returns the extent of an array along a specified dimension or the total number of elements in an array.

Argument: Array **Type:** Any

Result: I **Class:** I

Note:

1. Array must be an array. It must not be a pointer that is disassociated or an allocatable array that is not allocated. If Array is an assumed-size array Dim must be present with a value less than the rank of Array.

2. Dim must be scalar and in the range $1 \leq \text{Dim} \leq n$ where n is the rank of Array.
3. Result is equal to the extent of dimension Dim of Array, or if Dim is absent, the total number of elements of Array.

Example: A=SIZE(Array)

SPACING(X)

Returns the absolute spacing of model numbers near the argument value. See Chapter 8 and the real numeric model.

Argument: X **Type:** R

Result: As X **Class:** E

Example: S=SPACING(X)

SPREAD(Source,Dim,N_Copies)

Creates an array with an additional dimension, replicating the values in the original array.

Argument: Source **Type:** Any

Result: As Source **Class:** T

Note:

1. Source may be array valued or scalar, with rank less than 7.
2. Dim must be scalar and in the range $1 \leq \text{Dim} \leq n+1$ where n is the rank of Source.
3. N_Copies must be scalar.
4. The result is an array of rank $n+1$.

Example:

If A is the array (2,3,4) then SPREAD(A,DIM=1,NCOPIES=3) then the result is

the array $\begin{pmatrix} 2 & 3 & 4 \\ 2 & 3 & 4 \\ 2 & 3 & 4 \end{pmatrix}$

SQRT(X)

Square root.

Argument: X **Type:** R, C

Result: As argument **Class:** E

Example A=SQRT(B)

SUM(Array,Dim,Mask)

Returns the sum of all elements of Array along the dimension Dim corresponding to the true elements of Mask.

Argument: Array **Type:** N

Result: As Array **Class:** T

Note:

1. Array must be an array.
2. $1 \leq \text{Dim} \leq n$ where n is the rank of Array.
3. Mask must be conformable with Array.
4. Result is scalar if Dim is absent, or Array has rank 1, otherwise the result is an array of rank $n-1$.

Example:

1. SUM((/1,2,3/)) the result is 6.
2. SUM(C,Mask=C > 0.0) forms the arithmetic sum of the positive elements of C.

$$3. \text{ If } B = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$

SUM(B,Dim=1) is (3,7,11)

SUM(B,Dim=2) is (9,12)

SYSTEM_CLOCK(Count,Count_Rate,Count_Max)

Returns integer data from a real time clock.

Argument: Count **Type:** I

Result: N/A **Class:** S

Note:

1. Count is INTENT(OUT) and is set to a processor dependent value based on the current value of the processor clock or to -HUGE(0) if there is no clock. $0 \leq \text{Count} \leq \text{Count_Max}$.
2. Count_Rate is INTENT(OUT) and it is set to the number of processor clock counts per second, or zero if there is no clock.
3. Count_max is INTENT(OUT) and is set to the maximum value that Count can have or to zero if there is no clock.

Example: CALL SYSTEM_CLOCK(C,R,M)

TAN(X)

Tangent.

Argument: X **Type:** R

Result: As argument **Class:** E

Note: X must be in radians.

Example: Y=TAN(X)

TANH(X)

Hyperbolic tangent.

Argument: X **Type:** R

Result: As argument **Class:** E

Example: Y=TANH(X)

TINY(X)

Returns the smallest positive number in the model representing numbers of the same type and kind type parameter as the argument.

Argument: X **Type:** R

Result: As X **Class:** I

Example: T=TINY(X)

TRANSFER(Source,Mold,Size)

Returns a result with a physical representation identical to that of Source, but interpreted with the type and type parameters of Mold.

Argument: Source **Type:** Any

Result: As Mold **Class:** T

Warning: A thorough understanding of the implementation specific internal representation of the data types involved is necessary for successful use of this function. Consult the documentation that accompanies the compiler that you work with before using this function.

TRANSPOSE(Matrix)

Transposes an array of rank 2.

Argument: Matrix **Type:** Any

Result: As argument **Class:** T

Note: Matrix must be of rank 2. If its shape is (n,m) then the resultant matrix has shape (m,n) .

Example: For $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ `TRANPOSE(A)` yields $\begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$

TRIM(String)

Returns the argument with trailing blanks removed.

Argument: String

Type: S

Result: As String

Class: T

Note: String must be a scalar.

Example: `T_S=TRIM(S)`

UBOUND(Array,Dim)

Returns all the upper bounds of an array or a specified upper bound.

Argument: Array

Type: Any

Result: I

Class: I

Note:

$1 \leq Dim \leq n$, where n is the rank of Array. The result is scalar if Dim is present otherwise the result is an array of rank 1 and size n .

Result is a scalar if Dim is present otherwise is an array of rank 1, and size n .

Example: `Z=UBOUND(A)`

UNPACK(Vector,Mask,Field)

Unpacks an array of rank 1 into an array under the control of a mask.

Argument: Vector

Type: Any

Result: As Vector

Class: T

Note:

1. Vector must have rank 1. Its size must be at least t , where t is the number of true elements in Mask.

2. Mask must be array valued.

3. Field must be conformable with Mask. Result is an array with the same shape as Mask.

Example:

With $Vector = (1,2,3)$ and $Mask = \begin{pmatrix} F & T & F \\ T & F & F \\ F & F & T \end{pmatrix}$ and $Field = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

The result is $\begin{pmatrix} 1 & 2 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 3 \end{pmatrix}$

VERIFY(String,Set,Back)

Verify that a set of characters contains all the characters in a string by identifying the position of the first character in a string of characters that does not appear in a given set of characters.

Argument: String

Type: S

Result: I

Class: E

Note:

1. The default is to scan from the left, and will only be from the right when Back is present and has the value TRUE.
2. The value of the result is zero if each character in String is in Set, or if String has zero length.

Example: I=VERIFY(String,Set)